Efficiency Improvements in Constructing Pseudorandom Generators from One-way Functions

Working Draft: Please Do Not Distribute

Iftach Haitner* and Omer Reingold† and Salil Vadhan‡

April 9, 2010

Abstract

We give a new construction of pseudorandom generators from any one-way function. The construction achieves better parameters and is simpler than that given in the seminal work of Håstad, Impagliazzo, Levin, and Luby [SICOMP '99]. The key to our construction is a new notion of *next-block pseudoentropy*, which is inspired by the notion of "inaccessible entropy" recently introduced in [Haitner, Reingold, Vadhan, and Wee, STOC '09]. An additional advantage over previous constructions is that our pseudorandom generators are parallelizable and invoke the one-way function in a non-adaptive manner. Using [Applebaum, Ishai, and Kushilevitz, SICOMP '06], this implies the existence of pseudorandom generators in NC^0 based on the existence of one-way functions in NC^1.

Keywords: one-way function; pseudorandom generator; pseudoentropy

[Salil's Note: dropped "security preserving" since we don't discuss this in the paper anywhere. Do we get any qualitative improvement here (beyond what follows from the seed length improvement)?] [Iftach's Note: Don;t think so, but the seed length should dominate the other factors in most setting (at least this is what we had in mind when writing HHR)]

1 Introduction

The result of Håstad, Impagliazzo, Levin, and Luby [13] that one-way functions imply pseudorandom generators is one of the centerpieces of the foundations of cryptography and the theory of pseudorandomness.

From the perspective of cryptography, it shows that a very powerful and useful cryptographic primitive (namely, pseudorandom generators) can be constructed from the minimal assumption for complexity-based cryptography (namely, one-way functions). With this starting point, numerous

*Microsoft Research, New England Campus. E-mail: iftach@microsoft.com.

†Microsoft Research, Silicon Valley Campus, and Weizmann Institute of Science. E-mail:omreing@microsoft.com. Supported by US-Israel BSF grant 2006060.

‡School of Engineering and Applied Sciences and Center for Research on Computation and Society, Harvard University. E-mail: salil@seas.harvard.edu. Supported by NSF grant CNS-0831289 and US-Israel BSF grant 2006060.

other cryptographic primitives can also be constructed from one-way functions, such as private-key cryptography [5, 20], bit-commitment schemes [21], zero-knowledge proofs for NP [6], and identification schemes [3].

From the perspective of pseudorandomness, it provides strong evidence that pseudorandom bits can be generated very efficiently, with smaller computational resources than the "distinguishers" to whom the bits should look random. Such kinds of pseudorandom generators are needed, for example, for hardness results in learning [27] and the natural proofs barrier for circuit lower bounds [22]. Moreover, the paper of Håstad et al. introduced concepts and techniques that now permeate the theory of pseudorandomness, such as pseudoentropy and the Leftover Hash Lemma.

A drawback of the construction of Håstad et al., however, is that it is quite complicated. While it utilizes many elegant ideas and notions, the final construction combines these in a rather ad hoc and indirect fashion due to various technical issues. In addition to being less satisfactory from an aesthetic and pedagogical perspective, the complexity of the construction also has a significant impact on its efficiency. Indeed, it is too inefficient to be implemented even for very modest settings of parameters.

In the last few years, progress has been made on simplifying the construction of Håstad et al. [15] and improving its efficiency [9]. These constructions, however, still retain the overall structure of the Håstad et al. construction, and thus retain some of the complex and ad hoc elements.

In this paper, we present a significantly more direct and efficient construction of pseudorandom generators from one-way functions. The key to our construction is a new notion of *next-block pseudoentropy*, which is inspired by the recently introduced notion of "inaccessible entropy" [12].

1.1 The HILL Construction

Informally, a function $f : \{0,1\}^n \to \{0,1\}^n$ is a *one-way function* (OWF) if it is easy to compute (in polynomial time) and hard to invert even on random inputs. A polynomial-time computable function $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ is a *pseudorandom generator* (PRG) if it is stretching (i.e., $m(n) > n$) and its output distribution is pseudorandom (i.e., $G(U_n)$ is computationally indistinguishable from $U_{m(n)}$). The theorem of Håstad et al. relates these notions:

Theorem 1.1. *If there exists a one-way function, then there exists a pseudorandom generator.*

The key notion underlying their construction is the following generalization of pseudorandomness.

Definition 1.2 (pseudoentropy, informal)**.** *A random variable X has* pseudoentropy at least k *if there exists a random variable Y such that:*

1. *X is computationally indistinguishable from Y.*
2. *$\mathrm{H}(Y) \geq k$, where $\mathrm{H}(\cdot)$ denotes Shannon entropy.*[1]

A pseudoentropy generator (PEG)[2] *is a polynomial-time computable function $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ such that $X = G(U_n)$ has pseudoentropy at least $\mathrm{H}(G(U_n)) + \Delta(n)$ for some $\Delta(n) \geq 1/\mathrm{poly}(n)$. We refer to $\Delta(n)$ as the* entropy gap *of G.*

[1] The *Shannon entropy* of a random variable X is defined to be $\mathrm{E}_{x \xleftarrow{R} X}[\log(1/\Pr[X = x])]$.

[2] Håstad et al. [13] refer to such a generator as a *false entropy generator*, and require that a pseudoentropy generator to have output pseudoentropy (at least) $n + \Delta(n)$, rather than just $\mathrm{H}(G(U_n)) + \Delta(n)$. For the informal discussion here, however, we prefer not to introduce the additional term "false entropy".

[**Salil's Note: to do: I think it would be good to always say "(next-bit) pseudoentropy *at least* k" - this is a change we'd need to make throughout the paper**][**Iftach's Note: Done**]

That every pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ is a pseudoentropy generator can be seen by taking $Y = U_{m(n)}$ and noting that $\mathrm{H}(Y) = m(n)$, but $\mathrm{H}(G(U_n)) \leq \mathrm{H}(U_n) = n$. Pseudoentropy generators are weaker in that Y may be very far from uniform, and may even have $\mathrm{H}(Y) < n$ (as long as $\mathrm{H}(G(U_n))$ is even smaller).

The construction of pseudorandom generators from one-way functions proceeds roughly in the following steps:

OWF to PEG: Given a one-way function $f : \{0,1\}^n \to \{0,1\}^n$, Håstad et al. define $\mathrm{PEG}(x,h,i) = (f(x), h, h(x)_{1\ldots i})$, where h is an appropriate hash function and $h(x)_{1\ldots i}$ denotes the first i bits of $h(x)$. PEG can be shown to be a pseudoentropy generator with an entropy gap of roughly $\Delta = \log n/n$ — Whenever $i = \log|f^{-1}(x)| + \Theta(\log n)$ (which happens with probability $\Theta(\log n/n)$) the first $\approx \log|f^{-1}(x)|$ bits of $h(x)$ extract all the entropy of x, and then we get $\Theta(\log n)$ bits of pseudoentropy by the Goldreich–Levin Hardcore-Bit Theorem [4].

Converting Shannon Entropy to Min-Entropy and Amplifying the Gap: Next, Håstad et al. use a direct product construction $\mathrm{PEG}'(x_1,\ldots,x_t) = (\mathrm{PEG}(x_1),\ldots,\mathrm{PEG}(x_t))$ to convert pseudoentropy into pseudo-*min*-entropy, and increase the entropy gap to be $\omega(\log n)$. This turns out to require taking roughly $t = (n/\Delta)^2$ copies.

Randomness Extraction: By hashing, Håstad et al. extract pseudorandom bits from the pseudo-min-entropy achieved so far. By also hashing the seed x to extract any remaining entropy, they obtain a pseudorandom generator. Specifically, they show that $G(x,h_1,h_2) = (h_1,h_2,h_1(\mathrm{PEG}'(x)),h_2(x))$ is a pseudorandom generator, if the output lengths of h_1 and h_2 are chosen appropriately. The choice of output lengths depends on the amount of min-entropy in the output of PEG', which in turn depends on the amount of entropy in the output of PEG. Unfortunately, these quantities may be infeasible to compute; this is handled by the next step.

Enumeration: Håstad et al. enumerate over all $u = O(n/\Delta)$ possible values k for the output entropy of PEG (up to an accuracy of $\Delta/2$), construct a pseudorandom generator G_k for each, use composition to make each G_k stretch its seed by a factor of greater than u, and then take $G(x_1,\ldots,x_u) = G_1(x_1) \oplus \cdots \oplus G_u(x_u)$ as their final pseudorandom generator.

The total seed length in this informal description is roughly $n \cdot t \cdot u = O(n^4/\Delta^3) = O(n^7)$. In fact, we have been cheating a bit in order to present the construction in a more modular way than in [13]. (The issues we ignored have to do with the samplability of source Y in Definition 1.2.) [**Salil's Note: edited parenthetical remark. we should check what happens in ILL, as I always thought the ILL construction, was as above, and thus should give $O(n^7)$ for nonuniform security.**] The actual seed length in the main construction presented in [13] is of $O(n^{10})$ (and the construction involves additional complications). A construction of seed length $O(n^8)$ is outlined in [13], and has been formalized and proven in [15].

Above we see three main sources of inefficiency in the construction: (1) the entropy gap Δ being fairly small, (2) the conversion of Shannon entropy to min-entropy, and (3) enumerating guesses for the output entropy of the initial pseudoentropy generator. Haitner, Harnik, and Reingold [9] show

how to save a factor of n in the enumeration step (by constructing a pseudoentropy generator in which more is known about how the entropy is distributed) to obtain a seed length of $O(n^7)$, but still all of the steps remain.

A further complication in the construction of Håstad et al. is that the reductions demonstrating the correctness of the construction are much more complex for uniform adversaries. This aspect of the proof has recently been simplified and made much more modular via Holenstein's uniform hardcore lemma [14, 15].

In case the one-way function is secure against exponential running time ($2^{\Omega(n)}$) adversaries, Holenstein [15] showed how to reduce the seed length to $O(n^4 \cdot \omega(\log n))$ (or $O(n^5)$ to obtain a PRG with exponential security), which was then improved by Haitner et al. [10] to $O(n \cdot \omega(\log n))$ (or $O(n^2)$ to obtain a PRG with exponential security).[3]

1.2 Our Approach

Our construction is based on a generalization of the notion of a pseudoentropy generator. It is motivated by the well-known fact that the pseudorandomness of a random variable X is equivalent to each bit of X being indistinguishable from uniform given the previous ones [28]. That is, $X = (X_1, \ldots, X_n)$ is computationally indistinguishable from $U_n = (Y_1, \ldots, Y_n)$ if and only if for every i, $(X_1, X_2, \ldots, X_{i-1}, X_i)$ is computationally indistinguishable from $(X_1, X_2, \ldots, X_{i-1}, Y_i)$. It is thus natural to consider what happens if we require not that X_i be pseudorandom given the previous bits, but only that X_i has *pseudoentropy* given the previous bits. More generally, we can allow the X_i's to be blocks instead of bits.

Definition 1.3 (next-block pseudoentropy, informal)**.** *A random variable $X = (X_1, \ldots, X_m)$ has* next-block pseudoentropy at least *k if there exists a set of random variables $Y = \{Y_1, \ldots, Y_m\}$, each jointly distributed with X, such that:*

1. *For every i, $(X_1, X_2, \ldots, X_{i-1}, X_i)$ is computationally indistinguishable from $(X_1, X_2, \ldots, X_{i-1}, Y_i)$.*
2. *$\sum_i \mathrm{H}(Y_i | X_1, \ldots X_{i-1}) \geq k$.*

A next-block pseudoentropy generator *(NBPEG) is a polynomial-time computable function $G : \{0,1\}^n \to (\{0,1\}^\ell)^m$ such that $(X_1, \ldots, X_m) = G(U_n)$ has next-block pseudoentropy at least $\mathrm{H}(G(U_n)) + \Delta(n)$, where again $\Delta(n)$ is called the* entropy gap.

That is, in total, the bits of X "look like" they have k bits of entropy given the previous ones. Note that the case of 1 block ($m = 1$) amounts to the definition of a pseudoentropy generator. Also note that, when $m > 1$, allowing Y to be correlated with X in this definition is essential: for example if all the blocks of X are always equal to each other (and have noticeable entropy), then there is no way to define Y that is independent of X and satisfies the first condition.

With this notion, our construction proceeds as follows.

OWF to NBPEG: Given a one-way function f, we define $G(x, h) = (f(x), h, h(x)_1, h(x)_2, \ldots, h(x)_n)$, where $h : \{0,1\}^n \to \{0,1\}^n$ is an appropriate hash

[3] In more detail, Holenstein's construction generalizes [13] for OWFs of "any hardness", while Haitner et al. [10] take a totally different route (based on the"randomized iterate" of a function introduced by Goldreich et al. [7]) and obtain constructions based on exponentially hard OWFs, as well as on (unknown-)regular OWFs.

function, and $h(x)_i$ is the i'th bit of $h(x)$. Notice that this is the same as the construction of Håstad et al., except that we do not randomly truncate the output. At first, this seems problematic; by revealing all of $h(x)$, it becomes easy for an adversary to compute x, and thus the pseudoentropy of output equals its real entropy (i.e., we have *zero* entropy gap). We show, however, that it does indeed have *next-block pseudoentropy at least* $n + \log n$, which is even larger than the seed length of G. We have gained in two ways here. First, the entropy gap is now $\Delta = \log n$ instead of $\Delta = \log n/n$. Second, we know the total amount of entropy in the output (though not the amount contributed by the individual blocks). These two advantages improve the complexity and security of the rest of the construction. Furthermore, the fact that the next-block pseudoentropy is larger than the seed length simplifies the construction, as we do not need to extract any additional entropy from the seed.

Entropy Equalization: Here we use a technique from [12] to convert our knowledge about the total entropy (summed over all blocks) into knowledge about the entropy in the individual blocks. We evaluate G on $u = O(n/\Delta)$ independent seeds and concatenate the outputs, but randomly shifted by $i \stackrel{R}{\leftarrow} [n]$ coordinates. This increases our seed length and our entropy by a multiplicatives factor of approximately u, but now almost all the blocks have pseudoentropy at least the average pseudoentropy of the blocks of G.

Converting Shannon Entropy to Min-Entropy and Amplifying the Gap: This works the same as in [13]. Again we take roughly $t = O(n/\Delta)^2$ copies, but concatenate them within each block to obtain an m-block generator G'. Now each of the m blocks is indistinguishable from having high *min-entropy* conditioned on the previous ones. Thus, what we have is computational analogue of a *block source* [2], which are random sources in which each block has high min-entropy conditioned on the previous ones.

Randomness Extraction: For this step, we use a known method for block-source extraction [2, 29] and define $G(x, h) = (h, h(G'(x)_1), \ldots, h(G'(x)_m))$, where h is a universal hash function. Since we know how much pseudo-min-entropy is in each block, there is no difficulty in choosing the output length of h.

In total, our seed length is roughly $O(n \cdot u \cdot t) = O(n^4)$. For the case of exponentially hard one-way functions, we can obtain $\Delta = \Omega(n)$, and thus achieve seed $O(n \cdot \omega(\log n))$ matching [10] (but, unlike [10], our construction uses nonadaptive calls to the one-way function).

Note that our construction involves no "guessing" of entropies, neither in the construction of the initial NBPEG G, nor in an enumeration step at the end. While the entropy equalization "costs" the same (namely $u = O(n/\Delta)$) as enumeration did, it is actually doing more for us. Enumeration handled our lack of knowledge of a single entropy value (for which there were only $O(n/\Delta)$ choices), but here equalization is handling lack of knowledge for approximately n entropy values (one for each block), for which there are exponentially many choices. Moreover, enumeration required composing the pseudorandom generators to increase their stretch, resulting in a construction that is highly sequential and makes adaptive use of the one-way function. Our pseudorandom generators make nonadaptive use of the one-way function and are parallelizable (e.g., in NC^1), for getting pseudorandom generators with small stretch. Using Applebaum et al. [1], this implies the existence of pseudorandom generators in NC^0 based on the existence of one-way functions in NC^1.

1.3 Relation to Inaccessible Entropy

The notion of next-block pseudoentropy generators was inspired by the notion of *inaccessible entropy generators* in [12]. These are generators G that also produce m blocks $(x_1, \ldots, x_m)$ with the property that it is infeasible for an adversary to *generate* a sequence of blocks $(x_1, \ldots, x_m)$ that are consistent with the output of G in such a way that entropy of the individual blocks x_i is high (conditioned on the state of the adversary after generating the previous blocks). Thus, in a computational sense, the output of G has *low entropy*. For this reason, the notions of next-block pseudoentropy generators and inaccessible entropy generators seem to be dual to each other.

The initial construction of an inaccessible entropy generator in [12] is $G(x) = (f(x)_1, \ldots, f(x)_n, x)$, which is very similar to our construction of a next-block pseudoentropy generator except that there is no hashing and the bits of $f(x)$ instead of $h(x)$ are treated as separate blocks. This initial step is followed by entropy equalization and gap amplification steps that are exactly the same as the one we use (but analyzed with respect to dual notions). The final hashing step there (to construct statistically hiding commitment schemes) is more complex than ours and is necessarily interactive.

Interestingly, the notion of inaccessible entropy generator was introduced in an attempt to make the construction of statistically hiding commitment schemes from one-way functions "as simple" as the construction of pseudorandom generators from one-way functions, via manipulating notions of computational entropy. (The previous construction, from [11], was extremely complex.) In return, that effort has now inspired our simplifications and improvements to the construction of pseudorandom generators.

1.4 Paper Organization

Notations and definitions used through this paper are given in Section 2, where the new notion of a next-block pseudoentropy generator is formally defined in Section 3. In Section 4 we present our construction of next-block pseudoentropy generator from one-way functions, where in Section 5 we show how to use next-block pseudoentropy generators to get a pseudorandom generator. Finally, in Section 6 we use the above reductions to prove the main result of this paper.

2 Preliminaries

2.1 Random Variables

Let X and Y be random variables taking values in a discrete universe $\mathcal{U}$. We adopt the convention that when the same random variable appears multiple times in an expression, all occurrences refer to the same instantiation. For example, $\Pr[X = X]$ is 1. The *support* of a random variable X is $\mathrm{Supp}(X) := \{x : \Pr[X = x] > 0\}$. We write $\Delta(X, Y)$ to denote the *statistical difference* (a.k.a. variation distance) between X and Y, i.e.

$$\Delta(X, Y) = \max_{T \subseteq \mathcal{U}} |\Pr[X \in T] - \Pr[Y \in T]| .$$

If $\Delta(X, Y) \leq \varepsilon$ (respectively, $\Delta(X, Y) > \varepsilon$), we say that X and Y are ε-*close* (resp., ε-far).

2.2 Entropy Measures

We will refer to several measures of entropy in this work. The relation and motivation of these measures is best understood by considering a notion that we will refer to as the *sample-entropy*: For a random variable X and $x \in \mathrm{Supp}(X)$, we define the sample-entropy of x with respect to X to be the quantity

$$\mathrm{H}_X(x) := \log(1/\Pr[X = x]).$$

The sample-entropy measures the amount of "randomness" or "surprise" in the specific sample x, assuming that x has been generated according to X. Using this notion, we can define the *Shannon entropy* $\mathrm{H}(X)$ and *min-entropy* $\mathrm{H}_\infty(X)$ as follows:

$$\mathrm{H}(X) := \mathop{\mathrm{E}}_{x \xleftarrow{\mathrm{R}} X}[\mathrm{H}_X(x)]$$
$$\mathrm{H}_\infty(X) := \min_{x \in \mathrm{Supp}(X)} \mathrm{H}_X(x)$$

Flattening Shannon Entropy. It is well-known that the Shannon entropy of a random variable can be converted to min-entropy (up to small statistical distance) by taking independent copies of this variable.

Lemma 2.1. *1. Let X be a random variable taking values in a universe $\mathcal{U}$, let $t \in \mathbb{N}$, and let $\varepsilon > 0$. Then with probability at least $1 - \varepsilon - 2^{-\Omega(t)}$ over $x \xleftarrow{R} X^t$,*

$$|\mathrm{H}_{X^t}(x) - t \cdot \mathrm{H}(X)| \leq O(\sqrt{t \cdot \log(1/\varepsilon)} \cdot \log(|\mathcal{U}| \cdot t)).$$

2. Let X, Y be jointly distributed random variables where X takes values in a universe $\mathcal{U}$, let $t \in \mathbb{N}$, and let $\varepsilon > 0$. Then with probability at least $1 - \varepsilon - 2^{-\Omega(t)}$ over $(x, y) \xleftarrow{R} (X^t, Y^t) := (X, Y)^t$,

$$\left|\mathrm{H}_{X^t|Y^t}(x|y) - t \cdot \mathrm{H}(X|Y)\right| \leq O(\sqrt{t \cdot \log(1/\varepsilon)} \cdot \log(|\mathcal{U}| \cdot t)).$$

Proof. 1. For $x = (x_1, \ldots, x_t)$, we have $\mathrm{H}_{X^t}(x) = \sum_{i=1}^t \mathrm{H}_X(x_i)$. Thus, when $x \xleftarrow{\mathrm{R}} X^t$, $\mathrm{H}_{X^t}(x)$ is the sum of t independent random variables $\mathrm{H}_X(x_i)$, and thus we can obtain concentration around the expectation (which is $t \cdot \mathrm{H}(X)$) via Chernoff-Hoeffding Bounds. These random variables $\mathrm{H}_X(x_i)$ are not bounded (as is required to apply the standard Chernoff–Hoeffding Bound), but they are unlikely to be much larger than $O(\log |\mathcal{U}|)$. Specifically, for every $\tau > 0$ we have

$$\begin{aligned}\Pr_{x_i \xleftarrow{\mathrm{R}} X}[\mathrm{H}_X(x_i) \geq \log(|\mathcal{U}|/\tau)] &\leq \sum_{x_i \in \mathcal{U}: \mathrm{H}_X(x_i) \geq \log(|\mathcal{U}|/\tau)} \Pr[X = x_i] \\ &\leq |\mathcal{U}| \cdot 2^{-\log(|\mathcal{U}|/\tau)} \\ &= \tau.\end{aligned}$$

A Chernoff Bound for random variables with such exponentially vanishing tails follows from [26], and it says that the probability that the sum deviates from the expectation by at least $\Delta \cdot (\log(|\mathcal{U}|/\tau)) + 2\tau t$ is at most $\exp(-\Omega(\Delta^2/t)) + \exp(-\Omega(\tau t))$, provided $\tau \in [0, 1]$. An appropriate choice of $\Delta = O(\sqrt{t \log(1/\varepsilon)})$ and $\tau = \min\{1, O(\log(1/\varepsilon)/t)\}$ completes the proof. [**Salil's Note: to do: add some more details on how this follows from [26] and standard Chernoff**]

2. Similar, noting that $\mathrm{H}_{X^t|Y^t}(x|y) = \sum_{i=1}^{t} \mathrm{H}_{X|Y}(x_i|y_i)$.

□

2.3 One-way Functions

Definition 2.2. *Let $f\colon \{0,1\}^n \mapsto \{0,1\}^m$ be a polynomial-time computable function, where n is a security parameter and $m = m(n)$. For $T = T(n)$ and $\varepsilon = \varepsilon(n)$, we say that f is a* (T,ε)-one-way function *if for every probabilistic algorithm A running in time T and all sufficiently large n, we have:*

$$\Pr[A(Y) \in f^{-1}(Y)] \leq \varepsilon,$$

where the probability is taken over $Y = f(U_n)$ and the coin tosses of A. We say that f is a one-way function *if it is a $(p(n), 1/p(n))$-one-way function for every polynomial p.*

2.4 Pseudorandom Generators

Definition 2.3. *Let X be a random variable, depending on a security parameter n, and taking values in $\{0,1\}^m$, for $m = m(n)$. For $T = T(n)$ and $\varepsilon = \varepsilon(n)$, we say that X is* (T,ε)-pseudorandom *if for every probabilistic distinguisher D running in time T and all sufficiently large n, we have:*

$$|\Pr[D(X) = 1] - \Pr[D(U_m) = 1]| \leq \varepsilon.$$

A polynomial-time computable function $G\colon \{0,1\}^n \mapsto \{0,1\}^m$ with $m = m(n) > n$ is a (T,ε)-pseudorandom generator *if $G(U_n)$ is (T,ε)-pseudorandom.*

We say that X is pseudorandom *if it is $(p(n), 1/p(n))$-pseudorandom for every polynomial p. Similarly, G is a* pseudorandom generator *G if $G(U_n)$ is pseudorandom.*

3 Next-block Pseudoentropy

In this section we formally define the new notion of next-block pseudoentropy, for the cases of both Shannon entropy and min-entropy. The definitions will differ from the informal definition given in the introduction (Definition 1.3) in the following two ways, both of which are important for the treatment of uniform adversaries:

- We will require indistinguishability even against algorithms that have an oracle for sampling from the joint distribution (X, Y_i). (This enables us to show, using a hybrid argument, that pseudoentropy increases when we taking many independent copies of X. In the case of nonuniform adversaries, no oracle for sampling from (X, Y_i) is needed, as the samples can be nonuniformly hardwired into the adversary.)
- In order to achieve the first item, we will allow the random variables Y_i to depend on the distinguisher.

Similar issues arise for treating uniform adversaries with standard pseudoentropy.

Definition 3.1. *(next-block (Shannon) pseudoentropy) Let X be a random variable taking values in $\mathcal{U}^m$, where X, $\mathcal{U}$, and m may all depend on a security parameter n. For $T = T(n)$, $k = k(n)$ and $\varepsilon = \varepsilon(n)$, we say that X has* (T,ε) next-block pseudoentropy at least k *if for every oracle-aided distinguisher $D^{(\cdot)}$ of running time at most T, there exists a set of random variables $\{Y_1, \ldots, Y_m\}$ over $\mathcal{U}$ such that:*

1. $\sum_{i=1}^{m} \mathrm{H}(Y_i \mid X_1, \ldots, X_{i-1}) \geq k$, *and*

2. $\mathrm{E}_{i \xleftarrow{R} [m]} \left[\Pr[D^{O_{X,Y}}(X_1, \ldots, X_i) = 1] - \Pr[D^{O_{X,Y}}(X_1, \ldots, X_{i-1}, Y_i) = 1] \right] \leq L \cdot \varepsilon$, *where* $O_{X,Y}(i)$, *for* $i \in [m]$, *samples according to the joint distribution* (X, Y_i), *and* $L = L(n)$ *is a bound on number of calls made by* D *to* $O_{X,Y}$ *(including the challenge itself).*

We say that X *has* next-block pseudoentropy at least k, *if it has* $(p(n), 1/p(n))$*-next-block pseudoentropy at least* k *for every polynomial* p. *We say that* every block *of* X *has* (T, ε)*-next-block pseudoentropy at least* $\alpha = \alpha(n)$, *if condition (1) above is replaced with* $\mathrm{H}(Y_i \mid X_{1,\ldots,i-1}) \geq \alpha$ *for every* $i \in [m]$.

Note in comparing to the informal description given in the introduction, here we have omitted absolute values in the indistinguishability condition (Condition 2) above (and below). This is purely for technical convenience, as D can use $O((m/\varepsilon)^2)$ random samples from its oracle to test whether the (signed) advantages inside the expectation are positive or negative to within an accuracy of $\pm\varepsilon/2m$ and negate itself for some values of i in order to ensure a positive advantage of at least $L\varepsilon/2$.

Definition 3.2. *(next-block pseudo-min-entropy) Let* X *be a random variable taking values in* $\mathcal{U}^m$, *where* X, $\mathcal{U}$, *and* m *may all depend on a security parameter* n. *For* $T = T(n)$, $\alpha = \alpha(n)$, *and* $\varepsilon = \varepsilon(n)$, *we say that* every block *of* X *has* (T, ε)-next-block pseudo-*min*-entropy α, *if for every oracle-aided distinguisher* $D^{(\cdot)}$ *running in time at most* $T(n)$, *there exists a set of random variables* $\{Y_1, \ldots, Y_m\}$ *over* $\mathcal{U}$ *such that:*

1. $\mathrm{H}_\infty(Y_i \mid X_{1,\ldots,i-1}) \geq \alpha$, *and*

2. $\mathrm{E}_{i \xleftarrow{R} [m]} \left[\Pr[D^{O_{X,Y}}(X_1, \ldots, X_i) = 1] - \Pr[D^{O_{X,Y}}(X_1, \ldots, X_{i-1}, Y_i) = 1] \right] \leq L \cdot \varepsilon$, *where* $O_{X,Y}$ *and* L *is as in Definition 3.1.*

We say that every block *of* X *has* next-block pseudo-min-entropy α, *if every block of* X *has* $(p(n), 1/p(n))$*-next-block pseudo-min-entropy* α, *for every polynomial* p.

Unless explicitly stated otherwise, in the following sections we view a distribution over $\{0,1\}^t$ as a t-block distribution. When we refer to the next-block pseudoentropy properties of a function $G : \{0,1\}^n \to \{0,1\}^m$, these refer to the random variable $G(U_n)$.

Nice to have:

1. `non-uniform version of the above`

2. `define next-block indistinguishability`

4 One-way Functions to Next-block Pseudoentropy Generator

This section will show how to construct a next-block pseudoentropy generator ${G_{nb}}^f$ out of a one-way function $f \colon \{0,1\}^n \mapsto \{0,1\}^n$.